On Term Rewriting Systems Having a Rational Derivation

Antoine Meyer *

IRISA, campus de Beaulieu, Rennes
LIAFA, Université de Paris 7
Antoine.Meyer@liafa.jussieu.fr

Abstract. Several types of term rewriting systems can be distinguished by the way their rules overlap. In particular, we define the classes of prefix, suffix, bottom-up and top-down systems, which generalize similar classes on words. Our aim is to study the derivation relation of such systems (i.e. the reflexive and transitive closure of their rewriting relation) and, if possible, to provide a finite mechanism characterizing it. Using a notion of rational relations based on finite graph grammars, we show that the derivation of any bottom-up, top-down or suffix systems is rational, while it can be non recursive for prefix systems.

1 Introduction

Word rewriting systems are among the most general formalisms found in computer science to model word transformations. They generalize grammars, can represent the runs of finite automata, transducers, pushdown automata or even Turing machines. They can thus be considered as a unifying framework to compare all these heterogeneous formalisms. For instance, [6] proposes a homogeneous presentation of several well-known families of infinite graphs, using an approach based on word rewriting systems proposed in [4], which is to consider the 'Cayley graph' of a rewriting system. In another paper [5], a classification of word rewriting systems according to the way their rules overlap is established. It is proved that the derivation relations of four classes of systems are rational, which means that they can be generated by finite transducers. These systems called left, right, prefix and suffix, were later used in [6]. Any other class is shown to contain at least one system whose derivation is not rational.

The aim of this work is to extend these results from words to terms. To summarize, we will be interested in term rewriting systems whose derivation can be characterized by a finite mechanism. First of all, we have to specify which definition of rationality for relations on terms we intend to use, as several distinct notions already exist (see [13] for an overview). Unfortunately, none of them is as widely adopted as the standard one for words, as each relies on different characteristics of the word case, and serves a different purpose. In this

* This work has been supported in part by the European IST-FET project ADVANCE (contract No. IST-1999-29082).

paper, we will adopt a notion introduced in [14], which makes use of hyperedge replacement graph grammars. The reason for this choice is the close similarity between the way these grammars work, and the asynchronous mechanism of a word transducer. Then we extend the definitions of left, right, prefix and suffix systems to terms, yielding what we will call bottom-up, top-down, prefix and suffix systems, and investigate the rationality of their derivation relations. We also mention recognizability preservation properties for bottom-up, top-down and suffix systems.

Numerous works deal with term rewriting systems. Among the closest to our approach, we can mention for instance [10] and [15], which specifically investigate the recognizability preservation properties of term rewriting systems. Both papers study classes of systems which properly include the class of top-down systems, and prove that they preserve recognizability. However, the derivation relations of these systems are not rational (more generally, no finite representation of these relations is given). On the contrary, Dauchet and Tison extensively studied ground term rewriting systems, i.e. systems whose rules do not contain variables [8]. In particular, they proved that these systems have a decidable first order theory with reachability [9] by explicitely building their derivation relation. From another point of view, [12] and [7] investigated the geometric properties of transition graphs of ground systems and compared this family of graphs with respect to other well-known families. Note that by definition, ground systems are a special kind of suffix systems. Finally, we can mention the theme of symbolic model-checking, whose main idea is to represent regular sets of configurations by finite word automata and system transitions by rewrite rules or transducers (see for example [2]). This field is currently being extended to systems with richer topologies, like trees [1,3]. A central problem relevant to this method is to compute the set of configurations reachable in any number of steps when starting from a *regular* set of configurations (for instance a recognizable term language).

This paper is organized as follows: after recalling a few basic notions about trees, terms and recognizable languages, we present the notion of rationality for term relations introduced in [14]. In Section 4 we introduce term rewriting systems, and detail the four subclasses we consider. The last two parts present our results concerning the rationality of the derivations of top-down, bottom-up and suffix systems, as well as remarks concerning their preservation of recognizability.

2 Terms and Trees

Let $F = \bigcup_{n\geq 0} F_n$ be a finite ranked alphabet, each F_n being a set of *function* symbols of arity n (elements of F_0 are *constants*), and X be a finite set of *variable* symbols. The set $T(F, X)$ of finite first-order terms on F with variables in X is the smallest set including X and satisfying $f \in F_n \ \wedge \ t_1, \ldots, t_n \in T(F, X) \ \Rightarrow f t_1 \ldots t_n \in T(F, X)$. The set $T(F, X)^+$ of tuples of terms will be called the set of *term words*. A term word $t = (t_1, \ldots, t_n)$ is usually noted $t_1 \ldots t_n$, and $t(i)$ is used to denote t_i. The dimension of t is called its length and noted $|t|$ (here $|t| = n$). Term words containing no variable are called *ground*. The set of ground terms is

noted $T(F, \emptyset)$ or simply $T(F)$. The set of variables actually occurring in a term or term word t is $Var(t)$, and t is said *linear* if each of its variables occurs only once. If moreover t has n variables, it is called a *n-context*. The variables of a n-context are conventionally noted $\Box_1, \ldots \Box_n$. The set of n-contexts is denoted by $C_n(F)$, the set of all contexts by $C(F)$. A common operation on terms is *substitution*. A substitution is fully defined by a mapping from X to $T(F, X)$, and extended to a morphism as follows: we note $t\sigma$ the application of a substitution σ to a term word t, which is done by replacing every occurrence of each variable x occurring in t by the term $\sigma(x)$. The set of substitutions over F and X is noted $S(F, X)$. For any term word $s = s_1 \ldots s_n$ and when t is a n-context, we use $t[s]$ as a shorthand notation for the variable substitution $t\{\Box_i \mapsto s_i \mid i \in [1, n]\}$. All these notations are extended to sets of term words in the usual way. A term, term word, context or substitution is said to be *proper* or *non-trivial* if it contains at least one symbol in F.

Let $\mathbb{N}$ be the set of strictly positive integers, we call *position* any word in the set $\mathbb{N}^*$. Every term t in $T(F, X)$ can be represented as a finite ordered tree whose nodes are labeled by symbols in F or variables in X, or equivalently as a mapping from a prefix-closed set of *positions* $Pos(t)$, called the *domain* of the term, to the set $F \cup X$. Let $t = f(t_1, \ldots, t_n, \ldots)$ be a term represented by an ordered tree, position ε denotes the root of t, and for $n \in \mathbb{N},\ p \in \mathbb{N}^*$, position np denotes the node at position p in subtree t_n. Seeing terms as trees, term words can be seen as *ordered forests*. In the following, we will use the prefix partial order on positions, noted $\geqslant$: let p and q be two positions, $p \geqslant q$ if there is some $q' \in \mathbb{N}^*$ such that $p = qq'$. If furthermore $q' \neq \varepsilon$, we write $p > q$. We denote by $pos(x, t)$ the set of positions at which the variable $x \in X$ occurs in term $t \in T(F, X)$.

The most common acceptors for languages of trees (and thus terms) are finite tree automata. Among several variants, we will only consider *top-down* tree automata, defined by a finite set Q of control states and a finite set R of transition rules of the form $qf \to fq_1 \ldots q_n$ where $f \in F_n$ (n can be 0) and $q, q_1, \ldots q_n \in Q$. A configuration is an embedding of control states in the input tree, i.e. a tree from $T(F \cup Q, X)$, where each $q \in Q$ is considered a unary function symbol. A rule $qf \to fq_1 \ldots q_n$ can be applied in configuration c_1 to reach configuration c_2 if $c_1 = t[qft_1 \ldots t_n]$ and $c_2 = t[fq_1t_1 \ldots q_nt_n]$ for any context t and term word $t_1 \ldots t_n$. A run of the automaton is a sequence of applications of rules on a given input. A ground term $t \in T(F, X)$ is *accepted* or *recognized* if there is a run from configuration q_0t (where q_0 is an initial control state) to configuration t. The set of terms accepted by a tree automaton A is called the language of A and noted $L(A)$. The languages accepted by finite tree automata are called *recognizable*.

3 Rational Tree Relations

Several authors have tried to define suitable notions of binary relations over terms generalizing known families of relations over words, like for instance the

recognizable relations, or the more general rational relations (i.e. relations recognized by finite transducers). As of now, no extension to terms is really considered canonical, as each family of relations has its own merits and drawbacks. Several distinct families can be encountered: recognizable relations as such, relations defined as rational languages over some overlap coding of both projections of the relation, relations induced by various types of tree transducers, or the more specific class of ground tree transductions, to cite but a few (see [13] for a survey).

In [14], a notion of rationality for tuples of trees according to the union, substitution and iterated substitution operations is proposed. This notion can also be seen as a definition for binary rational relations over tuples of trees, and thus as a special case, binary relations over trees. Similarly to the word case, this class is strictly more general than the class of recognizable relations. In his paper, Raoult proves that the rational languages of tuples can be generated using a special kind of hyperedge replacement grammars. This definition is justified by its similarity to rational word relations on several aspects: first, as it should be, it coincides with rational word relations when restricted to trees of degree one. Second, it is closed under projection on any number of components, union and intersection. Finally, its mechanism is indeed quite close to the way a transducer works. However, this generality has a cost, and this class of relations is not closed under composition.

First, we need to define the product operation we shall use to define rational sets, which is an extension of the usual substitution operation. Let t be a term word, x a word of n variables having $k \geq 0$ instances $x_1 \dots x_k$ in t (i.e. a total of $n * k$ variables), and M a set of n-tuples of terms. We define $t \cdot_x M$ as the set of tuples of terms obtained by replacing each instance of x in t with a (possibly different) element of M. Formally: $t \cdot_x M := \{t\{s_i(j) \mapsto x_i(j) \,|\, i \in [1,k],\, j \in [1,n]\}\}$. It is extended to sets in the usual way: $L \cdot_x M := \{t \cdot_x M \mid t \in L\}$. Furthermore, define $L^{n_x} := L \cdot_x L^{n-1_x}$ and $L^{*_x} := \bigcup_{n \geq 0} L^{n_x}$. We are now ready to define the notion of rationality associated to this product:

Definition 3.1 ([14]). *The set Rat_n of rational languages of n-tuples of trees is the smallest set of languages containing the finite languages of tuples and closed under the following operations:*

1. $L \in Rat_n \ \wedge \ M \in Rat_n \ \Rightarrow \ L \cup M \in Rat_n$
2. $L \in Rat_n \ \wedge \ x \in X^m \ \wedge \ M \in Rat_m \ \Rightarrow \ L \cdot_x M \in Rat_n$
3. $L \in Rat_n \ \wedge \ x \in X^n \ \Rightarrow \ L^{*_x} \in Rat_n$

The family Rat of rational languages over tuples of terms is the union of all Rat_n, for $n \geq 1$.

One should note that this notion of rationality differs from the one defined in [11], for example, as the concatenation (or 'series') product is not directly taken into account, and substitution is done simultaneously on several variables. From this definition arises a straightforward notion of *rational expression*, which extends the usual notion on words. It should be noted that Rat_1 does not coincide with the set of recognizable term languages. For example, on

$\Sigma = \{f^{(2)}, g^{(1)}, h^{(1)}, a^{(0)}\}$, the language $fg^nag^na \in Rat_1$ is defined by the rational expression $f\Box_1\Box_2[g\Box_1 g\Box_2]^*[aa]$, but it is not a recognizable term language.

Let us now recall the hyperedge replacement grammars used in [14], which generate the rational languages of tuples of terms. In this paper, we will call *grammar* a hyperedge replacement grammar such that every production (A, α) has the following properties:

- the terminal subgraph of α, say α_t, obtained by removing all non-terminal hyperedges from α, is an ordered forest with n connex components (a n-tuple of trees), where n is the arity of A,
- the vertices of α belonging to a hyperedge are leaves of α_t,
- no vertex of α belongs to more than one hyperedge.

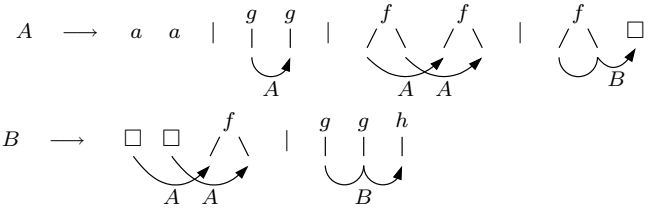
These properties allow us to refer to the right-hand sides of this type of grammars as 'leaf-linked forests'. The definition of grammar derivation is the usual one for hyperedge replacement. It will be useful to also recall the formal definition of a grammar from the point of view of terms, as it is done in the original paper:

Definition 3.2 ([14]). *Given a set X of variables, a* production *is a pair (A, α), where $A \in X^n$ ($A = A_1 \dots A_n$ is called a* non-terminal*), $\alpha \in T(F, X \times \mathbb{N})^n$ (here $X \times \mathbb{N}$ denotes the set of numbered instances of variables of X), and both A and α are linear. A* grammar *is a finite set of productions such that the variables occurring in the right-hand sides can be grouped to form instances of non-terminals. A step of derivation of a grammar is defined as $t \to_G t\{A^i_j \mapsto \alpha_j \mid j \in [1, n]\}$ where t is a term word, there is a production (A, α) in G and A^i is an instance of A in t. The language generated by a grammar G from axiom A is the set of tuples of ground trees $L(G, A) = \{w \in T(F)^{|A|} \mid A \to^*_G w\}$.*

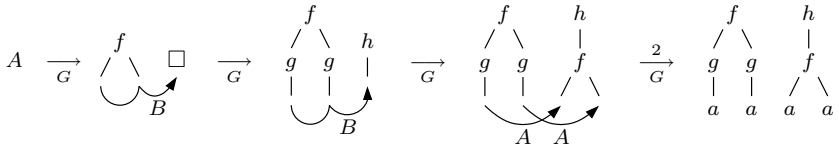
Example 3.3. Let $A = A_1A_2$ and $B = B_1B_2B_3$ be two non-terminals of respective arity 2 and 3. The grammar G_1 having rules

$$A \longrightarrow a\ a \mid gA_1\ gA_2 \mid fA^1_1A^2_1\ fA^1_2A^2_2 \mid fB_1B_2\ B_3$$
$$B \longrightarrow A^1_1\ A^2_1\ fA^1_2A^2_2 \mid gB_1\ gB_2\ hB_3$$

can be represented as a HR grammar in the following way:



Then a possible production sequence of G_1 would be:



As expected, these grammars generate the rational languages:

Theorem 3.4 ([14]). *A language of tuples of terms is rational if and only if it is the language generated by a grammar.*

A rational language of n-tuples of terms can also be seen as a binary relation in $T(F)^p \times T(F)^q$, where $p+q=n$. In this case, given a non-terminal A, we define the first and second projections $\pi_1(A)$ and $\pi_2(A)$ by the set of variables of A referring to the first (resp. second) projection of the relation. A similar notation is used for right-hand sides of grammar productions as well. For clarity, we write a production (A, α) as $(A, \pi_1(\alpha) \times \pi_2(\alpha))$. Without loss of generality, we always consider that $A = \pi_1(A)\pi_2(A)$ and $\alpha = \pi_1(\alpha)\pi_2(\alpha)$. For example, if the axiom of a $T(F)^p \times T(F)^q$ relation is $A = A_1 \dots A_n$, we can have $\pi_1(A) = A_1 \dots A_p$ and $\pi_2(A) = A_{p+1} \dots A_{p+q=n}$.

Example 3.5. Grammar G_1 from Ex. 3.3 generates, from non-terminal A, a language $L(G_1, A) \in Rat_2$, which can be seen as a $T(F) \times T(F)$ relation. In this case, its rules can be written

$$A \longrightarrow a \times a \mid gA_1 \times gA_1 \mid fA_1^1A_1^2 \times fA_2^1A_2^2 \mid fB_1B_2 \times B_3$$
$$B \longrightarrow A_1^1 \; A_1^2 \times fA_2^1A_2^2 \mid gB_1 \; gB_2 \times hB_3$$

4 Term Rewriting Systems

A (term) *rewrite rule* is a pair $(l, r) \in T(F, X)^2$ such that $Var(r) \subseteq Var(l)$. A rewrite rule (l, r) is said to be *linear* if both l and r are. A *rewrite system*, or more specifically *term rewriting system* is a set of rewrite rules R. A system R is *finite* when $|R|$ is finite, and *recognizable* when the potentially infinite number of rules is given as a finite union of pairs $U \to V$, where U and V are recognizable term languages. Note that we only consider systems where the total number of distinct variables is finite. A system is *linear* when all its rules are linear. We denote by $Dom(R)$ (resp. $Ran(R)$) the set of left-hand sides (resp. right-hand sides) of R, up to a renaming of the variables. The *rewriting* according to a system R is the relation

$$\underset{R}{\longrightarrow} := \{(c[l\sigma], c[r\sigma]) \in T(F) \times T(F) \mid (l, r) \in R \wedge c \in C_1(F) \wedge \sigma \in S(F, X)\}.$$

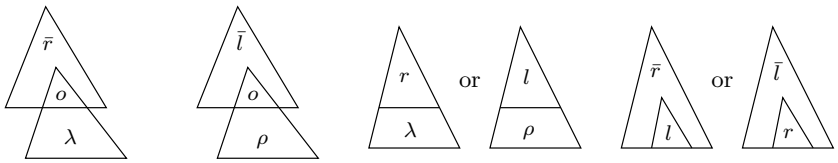
In case we want to specify that a rule (l, r) is used at some position p (resp. set of positions P), we use the notation $\underset{l,r}{\longrightarrow}_p$ (resp. $\underset{l,r}{\longrightarrow}_P$). The reflexive and transitive closure of $\to_R$ by composition is called the *derivation* of R and written $\to_R^*$.

Classification of Rewriting Systems. In the case of words, several natural classes of rewriting systems can be distinguished by the way their rules are allowed to overlap. In [5], the composition $\to_R \circ \to_R$ of two rewritings is considered, and all the different possibilities of overlapping between the right-hand side of

the first rewrite rule, and the left-hand side of the second one are examined. By discarding systems where unwanted overlappings occur, one obtains four general families of systems whose derivation is proven rational, the families of *left*, *right*, *prefix* and *suffix* word rewriting systems. Moreover, any system which does not belong to one of these families may have a non-rational derivation. As a consequence, as terms generalize words, we only need to study the extension of these four families of systems to terms: the classes of *bottom-up* and *top-down* systems, which respectively correspond to left and right systems, and the families of *prefix* and *suffix* systems. A term rewriting system R (resp. its inverse R^{-1}) is said:

- *top-down* (resp. *bottom-up*) if any overlapping between a right-hand side r and a left-hand side l of R (resp. R^{-1}) is such that $r = \bar{r}[o]$ and $l = o\lambda$ for some (possibly trivial) 1-context $\bar{r}$ and substitution λ,
- *prefix* if any overlapping between a right-hand side r and a left-hand side l of R is such that $l = r\lambda$ or $r = l\rho$ for some possibly trivial substitutions λ and ρ,
- *suffix* if any overlapping between a right-hand side r and a left-hand side l of R is such that $l = \bar{l}[r]$ or $r = \bar{r}[l]$ for some possibly trivial 1-contexts $\bar{r}$ and $\bar{l}$.

The following picture illustrates these four kinds of overlappings:



Prefix and suffix systems respectively generalize root and ground rewriting systems. Root rewriting systems are already known to be very powerful: indeed, they can simulate the execution steps of Turing machines. This implies a direct negative result concerning prefix systems.

Proposition 4.1. *Some linear prefix tree rewriting systems have a non rational derivation.*

Proof. Let M be a Turing machine with a set of states Q, a tape alphabet P and a set of transition rules $T \subseteq (Q \times P \cup \{\#\} \to Q \times P \times \{+, -\})$ ($\#$ denotes the 'blank' character). Let us build a prefix system R_M on the alphabet $Q \cup P \cup \{\#\}$, with variables in $\{x, y\}$, where Q is considered binary, P unary and $\#$ an overloaded symbol of arity either 0 or 1.
For all $pA \to qB+ \in T$, R_M has a rule $pxAy \to qBxy$, plus a rule $px\# \to qBx\#$ if $A = \#$.
For all $pA \to qB- \in T$ and $C \in P$, R_M has rules $pCxAy \to qxCBy$ and $p\#Ay \to p\#\#By$, plus rules $p\#\# \to q\#\#B\#$ and $pCx\# \to qxCB\#$ if $A = \#$.
This system has both overlappings of the kind $l = r\sigma$ and of the kind $r = l\sigma$, for

some left and right-hand sides l and r and substitution σ. It is thus prefix, and neither top-down, bottom-up or suffix in general. It is quite clear that computing the derivation of R_M is equivalent to computing the reachability relation of M, thus is undecidable. Hence $\rightarrow_R^*$ is non-recursive and can obviously not be rational. □

However, contrary to the case of words, where prefix and suffix systems are dual and share the same properties, the situation is different in the case of terms. The family of ground rewriting systems, which is a sub-family of suffix systems, has already been studied by several authors. In particular, Dauchet and Tison [8] showed that the derivations of ground systems can be recognized by a certain type of composite automata called *ground tree transducers* (GTT). Section 6 will use similar arguments in order to prove that, more generally, any suffix system has a rational derivation. The two remaining families of term rewriting systems we consider, namely top-down and bottom-up systems, are dual. The next section puts focus on top-down systems, but all the results extend to the bottom-up case (see Corollary 5.5).

5 Derivation of Bottom-Up and Top-Down Systems

This section focuses on the study of top-down term rewriting systems and their derivations. For any finite linear top-down system, a grammar of tuples of terms generating its derivation relation can be built, which implies that this relation is rational. Furthermore, from the shape of the grammar, we observe that the derivation of such a system preserves the recognizability of term languages. Dual results can be obtained for bottom-up systems: the derivation of a linear bottom-up system is rational, and the inverse image of a recognizable term language is still recognizable.

Let us first observe that top-down systems enjoy a kind of *monotonicity* feature. Any rewriting sequence of such systems is equivalent to a sequence where the successive rewriting steps occur at non-decreasing positions in the input term. We call this *top-down rewriting*. Let R be a term rewriting system, we define its top-down rewriting $\hookrightarrow_R^*$ by:

$$\overset{*}{\underset{R}{\hookrightarrow}} = \bigcup_{n\geq 0} \overset{n}{\underset{R}{\hookrightarrow}} \text{ with } \begin{cases} \overset{0}{\underset{R}{\hookrightarrow}} = Id_{T(F)} \\ \overset{n}{\underset{R}{\hookrightarrow}} = \bigcup_{p_1,\ldots,p_n} \underset{u_1,v_1}{\longrightarrow} p_1 \circ \ldots \circ \underset{u_n,v_n}{\longrightarrow} p_n \end{cases}$$

such that the rewriting positions do not decrease along indexes ($\forall i, j,\ i < j \Rightarrow \neg(p_j < p_i)$), and if two successive positions are equal then the second rewriting should not have a trivial left-hand side ($(p_i = p_{i-1}) \Rightarrow (u_i \notin X)$). This last condition means that, for instance, the sequence

$$c[l\sigma] \underset{l,r}{\longrightarrow} c[r\sigma] \underset{x,r'}{\longrightarrow} c[r'\{x \mapsto r\sigma\}]$$

is not top-down, because the second rule produces its right-hand side 'higher' than the first one. The rewriting steps should be swapped to obtain the top-down sequence

$$c[l\sigma] \underset{x,r'}{\longrightarrow} c[r'\{x \mapsto l\sigma\}] \underset{l,r}{\longrightarrow}_{pos(x,c[r'])} c[r'\{x \mapsto r\sigma\}].$$

The next lemma expresses the fact that, given any rewriting sequence of a top-down system, rewriting steps can always be ordered into an equivalent top-down sequence.

Lemma 5.1. *The relations of derivation and top-down derivation of any top-down term rewriting system R coincide: $\rightarrow_R^* = \hookrightarrow_R^*$.*

We are now ready to prove the rationality of the derivation of any top-down rewriting system. Using this property of top-down systems, it is possible to build a grammar which directly generates the derivation of any such system. This grammar mimics the way a rational word transducer works, using its control state to keep in memory a finite subterm already read or yet to produce.

Theorem 5.2. *Every finite linear top-down term rewriting system R has a rational derivation.*

Proof. Let R be a finite linear top-down system. We denote by O the set of all overlappings between left and right parts of rules of R:

$$O = \{\ t \in C_n(F,X) \mid \exists\ s \in C_1(F,X), \\ u \in T(F,X)^n, s[t] \in Ran(R)\ \wedge\ t[u] \in Dom(R)\ \}. \quad (1)$$

Remark that $\Box$ belongs to O. We will now build a grammar G whose language is exactly the derivation of R. Its finite set of non-terminals is $\{<*>\} \cup Q$, where $Q = \{\ <t> = <t>_1\ \ldots\ <t>_{n+1}\ \mid t \in O\ \cap\ C_n(F)\}$ and, for all $<t> \in Q$, $\pi_2(<t>)$ is a single variable. The production rules of G are of four types.
Type (1): $\forall\ f \in F_n$,

$$<\Box>\ \rightarrow\ f\ <\Box>_1^1 \ldots <\Box>_1^n\ \times\ f\ <\Box>_2^1 \ldots <\Box>_2^n$$
$$<*>\ \rightarrow\ f\ <*>^1 \ldots <*>^n$$

Type (2): $\forall\ t \in O \cap C_n(F),\ t[u] \in O \cap C_m(F)$,

$$<t>\ \rightarrow\ u[\pi_1(<t[u]>)]\ \times\ \pi_2(<t[u]>)$$

Type (3): $\forall\ t[u] \in O,\ t \in O \cap C_\ell(F)$ (necessarily $\{u_1, \ldots u_\ell\} \subseteq O$),

$$<t[u]>\ \rightarrow\ \pi_1(<u_1>) \ldots \pi_1(<u_\ell>) \times\ t[\pi_2(<u_1>) \ldots \pi_2(<u_\ell>)]$$

Type (4): $\forall\ (t[u], s[v]) \in R,\ v = v_1 \ldots v_\ell$ (necessarily $\{t, v_1, \ldots v_n\} \subseteq O$),

$$<t>\ \rightarrow\ u\sigma\ \times\ s[\pi_2(<v_1>) \ldots \pi_2(<v_\ell>)]$$

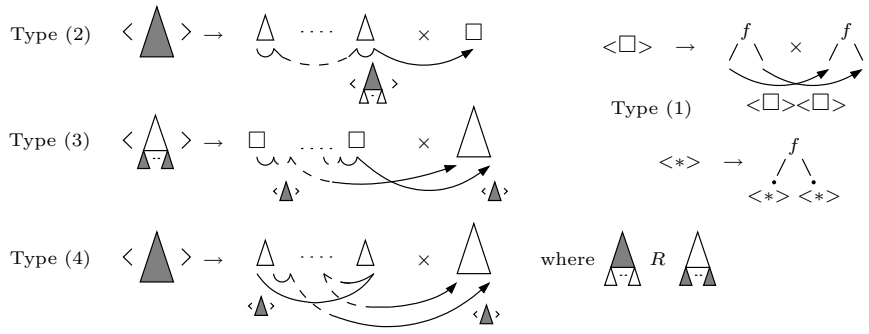


Fig. 1. grammar associated to a top-down system.

where σ is a variable renaming such that for any variable x of u, $\sigma(x) = <v_i>_j$ if x is the j-th variable to appear in v_i (from left to right), and $\sigma(x) = <*>$ if x does not appear in any of the v_i. Figure 1 illustrates the four types of rules.

Intuitively, the role of this substitution is to gather into the same non-terminal or hyperedge all the variables of u belonging to the same v_i, while respecting the order in which these variables appear in v_i. This way, a correct instantiation of non-terminals of G is ensured. If a variable of u does not appear at all in v, then it means that a whole input subtree is 'discarded' by the rewriting rule being applied. Thus the grammar should accept any subtree to be generated at this position, which is the role of the unary non-terminal $<*>$.

For simplicity, we will only consider type (4) rules in which $t, v_1, \dots v_n$ are *maximal*. The other cases can be simulated by suitable finite compositions of rules of types (2), (3) and (4). □

Example 5.3. Consider the linear top-down system R over the alphabet $F = \{f^{(2)},\ g^{(1)}, h^{(1)}, a^{(0)}\}$ with a unique rule $fgxgy \to hfxy$. The corresponding grammar is the grammar of Ex. 3.5 where each non-terminal stands for one of the possible overlappings of rules of R: A stands for $\Box$ and B for $f\Box_1\Box_2$. Note that type (4) rules with non-maximal overlappings have been discarded. This example also illustrates the fact that the inverse image of a recognizable term language by the derivation of a linear top-down system is not recognizable in general: for instance, the image by R_G^{-1} of h^*faa is $\{h^*fg^nag^na \mid n \geq 0\}$, which is not recognizable.

We will now mention a property of top-down systems, which has been known for the past few years for larger classes of systems.

Proposition 5.4. *The image of any recognizable term language by the derivation relation of a finite linear top-down term rewriting system is recognizable.*

Top-down systems form a strict subfamily of *generalized semi-monadic term rewriting systems* [10], which is itself a strict subfamily of *right-linear finite path*

overlapping systems [15]. Both classes have been proven to preserve recognizability. As a consequence, this is also the case for top-down systems. However, it should be mentioned that neither of these classes has a rational derivation. Indeed, it is quite easy to find a generalized semi-monadic system whose derivation cannot be recognized by any finite mechanism. For instance, the generalized semi-monadic system whose unique rule is $gx \to fgfx$ clearly has a non-rational derivation: its intersection with the rational relation $ga \times f^*gf^*a$ is $ga \times \{f^ngf^na | n \geq 0\}$. By the usual pumping arguments (adapted to this new setting), this relation is not rational.

Finally, please note that the inverse of a top-down system is, by definition, bottom-up. For any top-down system we can build a grammar G recognizing $\to_R^*$. Thus, the grammar $\pi_2(G)\pi_1(G)$ obtained by swapping both projections of G generates the derivation $\to_{R^{-1}}^*$ of the bottom-up system R^{-1}. Inverse recognizability preservation follows.

Corollary 5.5. *Every finite linear bottom-up term rewriting system R has a rational derivation $\to_R^*$, and the inverse image by $\to_R^*$ of any recognizable term language is recognizable.*

6 Derivation of Suffix Systems

This section presents a study of the derivation relations of suffix term rewriting systems. After introducing a property related to the notion of *suffix rewriting*, we show that the derivation of any recognizable linear suffix system is rational. Finally, we prove that the image or inverse image of any recognizable term language by the derivation of a recognizable linear suffix system is recognizable, and that it is possible to build a tree automaton accepting it.

Definition 6.1. *The suffix rewriting of a term rewriting system R is the relation*

$$\xrightarrow[R]{} = \big\{ (c[l\sigma], c[r\sigma]) \in T(F,X)^2 \mid (l,r) \in R \ \wedge \ c \in C_1(F) \\ \wedge \ \sigma \in S(\emptyset, X) \ \textit{bijective} \big\}$$

(a bijective substitution in $S(\emptyset, X)$ is a bijective variable renaming over X).

Suffix systems have a specific behaviour with respect to suffix rewriting. Indeed, the derivation of any input tree t by a suffix system can always be decomposed in two phases. First, a prefix $\bar{t}$ of t is read, and several steps of suffix rewriting can be applied to it. Once this first sequence is over, $\bar{t}$ has been rewritten into a prefix $\bar{s}$ of s, never to be modified anymore. In a second time, the rest of t is derived in the same fashion, starting with suffix rewriting of a prefix of the remaining input. As a consequence, the derivation of a suffix system is equivalent to its 'iterated' suffix derivation.

Lemma 6.2. *For any suffix term rewriting system R,*

$$s \xrightarrow[R]{+} t \iff \begin{cases} \exists\ \bar{s}, \bar{t} \in T(F,X),\ \sigma, \tau \in S(F,X)\ s = \bar{s}\sigma\ \wedge\ t = \bar{t}\tau \\ \wedge\ \bar{s} \xrightarrow[R]{+}\!\!\!\!\rightarrow \bar{t}\ \wedge\ \forall x \in Var(\bar{s}) \cap Var(\bar{t}),\ \sigma(x) \xrightarrow[R]{*} \tau(x). \end{cases}$$

Another interesting property is that, for any recognizable system, a suffix rewriting sequence is always equivalent to a sequence in two parts, where the first part only consumes suffix subterms of the input term, and the second part only produces new suffix subterms in their place.

Lemma 6.3. *For all recognizable linear term rewriting system R over F and X, there exist a finite ranked alphabet Q and three finite rewriting systems*

- $R_- \subseteq \{px \to fp_1x_1 \dots p_nx_n \mid f \in F,\ p, p_1, \dots, p_n \in Q,\ x, x_1, \dots, x_n \in X^*\}$
- $R_= \subseteq \{px \to qy \mid p, q \in Q,\ x, y \in X^*\}$
- $R_+ \subseteq \{fp_1x_1 \dots p_nx_n \to px \mid f \in F,\ p, p_1, \dots, p_n \in Q,\ x, x_1, \dots, x_n \in X^*\}$

such that $s \xrightarrow[R]{*}\!\!\!\!\rightarrow t \iff s \xrightarrow[R_+ \cup R_=]{*}\!\!\!\!\rightarrow \circ \xrightarrow[R_- \cup R_=]{*}\!\!\!\!\rightarrow t$.

Lemma 6.3 can be reformulated in the following way: a pair (s,t) of terms belongs to the suffix derivation of a system R if and only if there is a context c such that $s = c[s_1 \dots s_n]$, $t = c[t_1 \dots t_n]$ and for all $i \in [1,n]$, there is a term q_ix_i such that $s_i \to\!\!\!\!\rightarrow^*_{R_+ \cup R_=} q_ix_i$ and $q_ix_i \to\!\!\!\!\rightarrow^*_{R_- \cup R_=} t_i$.

Theorem 6.4. *Every recognizable linear suffix term rewriting system R has a rational derivation.*

Proof. Let R be a recognizable linear suffix system on $T(F,X)$. Let R_+, $R_=$ and R_- be the rewriting systems mentioned in Lemma 6.3. Let N be a set of pairs of the form $u|v$ where u and v are two linear term words over $Ran(R_+ \cup R_=)^*$ and $Dom(R_- \cup R_=)^*$ respectively. Note that Ran and Dom are defined up to a renaming of the variables. We can thus impose that u and v share the same set of variables ($Var(u) = Var(v)$), and there is no pair of strict subwords u' and v' of u and v such that $Var(u') \neq Var(v')$ (i.e one should not be able to split $u|v$ in two correct non-terminals). This, together with the facts that F is finite and u and v are linear, implies that N is finite for some fixed, standard variable renaming. Thus, given an axiom I, we can build a grammar G whose set of non-terminals is $N \cup \{I, I'\}$, having the following finite sets of productions:

$$\forall\ f \in F, \qquad I \longrightarrow fI^1_1 \dots I^n_1 \times fI^1_2 \dots I^n_2 \quad \text{and} \quad I' \longrightarrow fI'^1 \dots I'^n \tag{2}$$

$$\forall\ px \in Dom(R_- \cup R_=) \cap Ran(R_+ \cup R_=), \qquad I \longrightarrow px|px \tag{3}$$

$$\forall\ u' \to^*_{R_=} u,\quad v \to^*_{R_=} v',\quad u' \in Ran(R_+)^*,\quad v' \in Dom(R_-)^*,$$
$$u|v \ \longrightarrow\ u'|v' \quad (4)$$

$$\forall\ u_1 = p_1x_1 \dots p_ix_i,\quad u_2 = p_{j+1}x_{j+1} \dots p_nx_n,$$
$$v = q_1y_1 \dots q_my_m,\quad f p_{i+1}x_{i+1} \dots p_jx_j\ R_+\ px,$$
$$u_1\, px\, u_2|v \ \longrightarrow\ \mu_1 \dots \mu_i\,(f\mu_{i+1} \dots \mu_j)\,\mu_{j+1} \dots \mu_n\ \times\ \nu_1 \dots \nu_m \quad (5)$$

$$\forall\ u = p_1x_1 \dots p_nx_n,\quad v_1 = q_1y_1 \dots q_iy_i,$$
$$v_2 = q_{j+1}y_{j+1} \dots q_my_m,\quad qy\ R_-\ f q_{i+1}y_{i+1} \dots q_jy_j,$$
$$u|v_1\, qy\, v_2 \ \longrightarrow\ \mu_1 \dots \mu_n\ \times\ \nu_1 \dots \nu_i\,(f\nu_{i+1} \dots \nu_j)\,\nu_{j+1} \dots \nu_m \quad (6)$$

In rules (5) and (6), all the $(\mu_k)_{k\in[1,n]}$ and $(\nu_k)_{k\in[1,m]}$ are variables belonging to instances of non-terminals $u'|v' \in N$ where u' and v' are built from terms $(p_kx_k)_{k\in[1,n]}$ and $(q_ky_k)_{k\in[1,m]}$ respectively. Variables μ_1 to μ_n (resp. ν_1 to ν_m) appear only in the first (resp. second) projection of any non-terminal. Note that this instantiation is unique, by construction of the set N. It is also always possible since every rule of R is, by hypothesis, linear.

Call ρ the substitution which maps each non-terminal variable $(u|v)_i$ to the term $(u)_i$ if $i \in [1, |u|]$ and to $(v)_i$ if $i \in [|u|+1, |u|+|v|]$, and each non-terminal variable $(I^i_j)_{j\in[1,2]}$ to a variable x_i. It is clear from the rules of G_0 that:

$$I \xrightarrow[G_0]{*} s \times t \iff s\rho \xrightarrow[R_+\cup R_=]{*} \circ \xrightarrow[R_-\cup R_=]{*} t\rho. \quad (7)$$

We will not detail the proof of this observation. Notice that this grammar works in a very similar way to a *ground tree transducer*, which is the formalism used by [8] to recognize the derivation of a ground system. The only difference is that we keep track of the variables appearing in the left and right projections of the relation, so as to be able to resume the rewriting at relevant positions. Now add to G_0 the set of rules

$$\forall\ x \in X \text{ such that } xRpx,\ qxRx, \qquad px|qx \ \longrightarrow\ I \quad (8)$$
$$px| \ \longrightarrow\ I' \quad (9)$$

and call this new grammar G. These last rules allow the derivation to go on properly after a first sequence of suffix rewritings has taken place, by creating new instances of the axiom between leaves where the same variable would appear. By Lemma 6.2, G generates $\to^*_R$. □

Proposition 6.5. *The image and inverse image of any recognizable term language by the derivation of a finite linear suffix term rewriting system is recognizable.*

Proof sketch. Once a grammar generating the derivation of a suffix system R is built, according to the previous proof, it is not difficult to synchronize the left projection of this grammar with any finite top-down tree automaton A. We

thus obtain a new grammar, whose second projection yields a finite automaton accepting the image of $L(A)$ by $\rightarrow_R^*$. This is symmetrical, hence the converse. □

We will illustrate the fact that suffix systems are strictly more general than ground systems on the following simple example.

Example 6.6. Consider the finite suffix system $R = \{fxy \rightarrow fyx,\ a \rightarrow ga\}$ over the ranked alphabet $\{f^{(2)},\ g^{(1)},\ a^{(0)}\}$. The first rule of R allows to swap at any time both children of an f-node. This somehow expresses the commutativity of f. The derivation of R (restricted for the sake of clarity to $(fg^*ag^*a)^2$) is the relation $\{(fg^mag^na, fg^nag^ma) \mid m, n \geq 0\} \cup \{(fg^mag^na, fg^{m+1}ag^na) \mid m, n \geq 0\} \cup \{(fg^mag^na, fg^mag^{n+1}a) \mid m, n \geq 0\}$, which is not recognizable by a ground tree transducer.

Furthermore, we claim that the transition graph of this rewriting system is not isomorphic to the transition system of any (recognizable) ground term rewriting system as defined in [12,7]. *Note:* the transition graph of a rewriting system is the graph whose vertices are the terms from the domain or range of the system, and whose edges are all the pairs (s, t) such that s can be rewritten to t in one step.

7 Conclusion

This paper extends the left, right, prefix and suffix word rewriting systems defined in [5] to bottom-up, top-down, suffix and prefix term rewriting systems. The derivation relation of the three first types of systems can be generated by finite graph grammars, while systems of the fourth type have a non recursive derivation in general. We also stated some recognizability preservation properties of these classes of systems, and provided effective constructions in each case. Although [15] defines a class of recognizability-preserving rewriting systems strictly more general than top-down systems, they do not aim to provide a construction for the derivation relation itself, which is indeed not rational. As for suffix systems, to our knowledge, no comparable class of recognizability-preserving systems has been defined yet.

This study puts in practical use the notion of rationality defined in [14], which nicely extends the usual rational relations on words, even though some of their key properties are missing, like the closure by composition or systematic preservation of recognizability. However, this formalism is an interesting and powerful work basis for the study of binary relations on terms, especially thanks to the fact that it is general enough to extend asynchronous transducers (which is not the case of most other formalisms). Still, depending on one's objectives, it might be necessary to devise a more restricted notion of rational relations on terms, which would be closed under composition or preserve recognizability (or both). Note that [14] contains the definition of such a subfamily of relations (called rational *transductions*). However, it can be shown that the derivations of some top-down systems do not belong to this class.

Finally, it could be interesting to look for extensions to some of the existing works previously mentioned. First, one may try to elaborate actual verification

methods using our systems to model transitions, and recognizable term languages for sets of configurations, along the ideas of regular model-checking [2]. Indeed, being able to effectively build the transitive closure of the system's transition relation and compute the image of regular sets of configurations could lead to interesting results. Second, the definitions from [12] and [7] about transition graphs of ground systems, should extend smoothly to the case of suffix systems. Thus, it would be meaningful to determine whether part or all of their results extend to this new family, and in particular whether the transition graphs of suffix systems have a decidable first order theory with reachability. Note that, as illustrated in Ex. 6.6, we suspect that the transition graphs of suffix systems *strictly* include the former families of graphs.

References

1. P. Abdulla, B. Jonsson, P. Mahata, and J. D'Orso. Regular tree model checking. In *CAV 14th*, volume 2404 of *LNCS*, pages 555–568, 2002.
2. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *CAV 12th*, volume 1855 of *LNCS*, pages 403–418, 2000.
3. A. Bouajjani and T. Touili. Extrapolating tree transformations. In *CAV 14th*, volume 2404 of *LNCS*, pages 539–554, 2002.
4. H. Calbrix and T. Knapik. A string-rewriting characterization of Muller and Schupp's context-free graphs. In *FSTTCS 18th*, volume 1530 of *LNCS*, pages 331–342, 1998.
5. D. Caucal. On word rewriting systems having a rational derivation. In *FoSSaCS 3rd*, volume 1784 of *LNCS*, pages 48–62, 2000.
6. D. Caucal and T. Knapik. A Chomsky-like hierarchy of infinite graphs. In *MFCS 27th*, volume 2420 of *LNCS*, pages 177–187, 2002.
7. T. Colcombet. On families of graphs having a decidable first order theory with reachability. In *ICALP 29th*, volume 2380 of *LNCS*, pages 98–109, 2002.
8. M. Dauchet and S. Tison. Decidability of the confluence of finite ground term rewrite systems. In *FCT 5th*, volume 199 of *LNCS*, pages 80–89, 1985.
9. M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *LICS 90th*, pages 242–248. IEEE, 1990.
10. P. Gyenizse and S. Vágvölgyi. Linear generalized semi-monadic rewrite systems effectively preserve recognizability. *TCS*, 194(1–2):87–122, 1998.
11. K. Lodaya and P. Weil. Rationality in algebras with a series operation. *Information and Computation*, 171(2):269–293, 2001.
12. C. Löding. Ground tree rewriting graphs of bounded tree width. In *STACS 19th*, volume 2285 of *LNCS*, pages 559–570, 2002.
13. J.-C. Raoult. A survey of tree transductions. Technical Report 1410, Inria, April 1991.
14. J.-C. Raoult. Rational tree relations. *Bulletin of the Belgian Mathematics Society*, 4:149–176, 1997.
15. T. Takai, Y. Kaji, and H. Seki. Right-linear finite path overlapping term rewriting systems effectively preserve recognizability. In *RTA 11th*, volume 1833 of *LNCS*, pages 246–260, 2000.

A Appendix: Proof details

We will present here the missing proof to Lemma 5.1, a construction which can be used as an alternative proof for Proposition 5.4, the proofs to Lemmas 6.2 and 6.3 and a detailed construction for Proposition 6.5.

A.1 Derivation of Bottom-up and Top-down Systems

Lemma 5.1. *The relations of derivation and top-down derivation of any top-down term rewriting system R coincide:*

$$\xrightarrow[R]{*} \;=\; \overset{*}{\hookrightarrow}_R.$$

Proof. By definition, $\hookrightarrow_R^* \subseteq \to_R^*$. It remains to prove the converse, by using the same technique as in [5]. As $\to_R^* = \to_{R-(X\times X)}^*$, we may assume that $R \cap (X \times X) = \emptyset$. Then, we can sort any derivation into a *top-down* derivation, as defined above, by applying an alternative version of the bubble sort algorithm in which removal and addition of elements are allowed. To do so, we use the following inclusions:

$$\xrightarrow[u,v]{}_{\geqslant p} \circ \xrightarrow[x,v']{}_{p} \subseteq \xrightarrow[x,v']{}_{p} \circ \xrightarrow[u,v]{}_{2^{>p}} \tag{10}$$

$$\xrightarrow[u,v]{}_{>p} \circ \xrightarrow[u',v']{}_{p} \subseteq \xrightarrow[u',v']{}_{p} \circ \xrightarrow[u,v]{}_{2^{\geqslant p}} \quad \text{with } u, u' \notin X \tag{11}$$

$$\xrightarrow[u,v]{}_{>p} \circ \xrightarrow[x,v']{}_{p} \subseteq \xrightarrow[x,v']{}_{p} \circ \xrightarrow[u,v]{}_{2^{>p}} \quad \text{with } u', v' \notin X \tag{12}$$

$$\xrightarrow[u,v]{}_{>p} \circ \xrightarrow[x,v']{}_{p} \subseteq \xrightarrow[x,v']{}_{p} \circ \xrightarrow[u,v]{}_{2^{>p}} \cup \xrightarrow[u,v]{}_{p} \circ \xrightarrow[x,v']{}_{2^{>p}} \tag{13}$$

where $\to_{>p}$ (resp. $\to_{\geqslant p}$) denotes rewriting at any position greater than (resp. greater or equal to) p, and $\to_{2^{>p}}$ or $\to_{2^{\geqslant p}}$ denotes multi-step rewriting at any set of such positions. Let us now prove the first inclusion. Let

$$r \xrightarrow[u,v]{}_{p} s \xrightarrow[x,v']{}_{q} t$$

with $p \geqslant q$. One can find 1-contexts $\bar{r}$ and $\bar{s}$ and substitutions ρ and σ such that $r = \bar{r}[u\rho]$, $s = \bar{r}[v\rho] = \bar{s}[x\sigma]$ and $t = \bar{s}[v'\sigma]$, with $pos(\Box, \bar{r}) = p$ and $pos(\Box, \bar{s}) = q$. As $p \geqslant q$ and by hypothesis on R, there is a substitution γ such that $\bar{r} = \bar{s}[x\gamma]$ and $\sigma = \gamma[v\rho]$. It is thus possible to swap the rewriting steps between r and t:

$$r = (\bar{s}[x\gamma])[u\rho] \xrightarrow[x,v']{}_{q} (\bar{s}[v'\gamma])[u\rho] \xrightarrow[u,v]{}_{P} (\bar{s}[v'\gamma])[v\rho] = t,$$

where $P \subseteq 2^{>q}$ is the set of positions at which the special variable $\Box$ occurs in $\bar{s}[v'\gamma]$ ($P = pos(\Box, \bar{s}[v'\gamma])$). Hence

$$r \xrightarrow[x,v']{}_{q} \circ \xrightarrow[u,v]{}_{2^{>q}} t,$$

and inclusion (10) is proven. More generally, consider any two-steps rewriting

$$r \underset{u,v}{\longrightarrow}_p s \underset{u',v'}{\longrightarrow}_q t$$

with $p > q$. By definition there must exist 1-contexts $\bar{r}$ and $\bar{s}$ and substitutions ρ and σ such that $r = \bar{r}[u\rho],\ s = \bar{r}[v\rho] = \bar{s}[u\sigma]$ and $t = \bar{s}[v'\sigma]$, with $pos(\Box, \bar{r}) = \{p\}$ and $pos(\Box, \bar{s}) = \{q\}$. The hypotheses we made on R imply certain restrictions on the structure of configuration s. As R is descending, $\bar{s}[u'] \leqslant \bar{r}$. So there must be a substitution γ such that $\bar{r} = \bar{s}[u'\gamma]$ and $\sigma = \gamma[v\rho]$. Hence

$$r = (\bar{s}[u'\gamma])[u\rho] \underset{u',v'}{\longrightarrow}_q (\bar{s}[v'\gamma])[u\rho] \underset{u,v}{\longrightarrow}_P (\bar{s}[v'\gamma])[v\rho] = t,$$

where $P = pos(\Box, \bar{s}[v'\gamma])$. Thus

$$r \underset{x,v'}{\longrightarrow}_q \circ \underset{u,v}{\longrightarrow}_P t.$$

As $P \in 2^{\geqslant q}$, inclusion (11) is proven. If $v' \notin X$ or $v' = y \in X \wedge \gamma(y)$ is not trivial, then $P \in 2^{>q}$, so (12) and the first part of (13) are true. Finally, if $u \in X,\ v' \in X$ and $\gamma(v')$ is trivial, then $r = \bar{r}\rho,\ t = \bar{s}\sigma,\ \bar{r} = \bar{s}[u'],\ v\rho = \sigma$, hence

$$r = (\bar{s}[u'])\rho \underset{x,v}{\longrightarrow}_q (\bar{s}[v[u']])\rho \underset{u',x'}{\longrightarrow}_P \subseteq 2^{>q} (\bar{s}[v])\rho = t.\Box$$

Proposition 5.4. *The image of any recognizable term language by the derivation of a linear top-down term rewriting system is recognizable.*

Proof. First note that the domain of the derivation of a descending system is $T(F)$, and its range is recognizable: the grammar constructed in the proof of Theorem 5.2 only has non-terminals whose second projection is reduced to a single variable.

Let L be a recognizable term language accepted by some top-down tree automaton A with a set of control states Q. Let R be a linear descending system. By the previous construction, we are able to build a rational grammar G recognizing $\rightarrow_R^*$. Let us now define a "product grammar" G_A whose domain is L and whose range is $\rightarrow_R^* (L)$. G_A's non-terminals will be of the form

$$(N_1, q_1) \dots (N_n, q_n)(N_{n+1}, q_1 \dots q_n), \tag{14}$$

noted $N_{q_1 \dots q_n}$ for short. For each production

$$N_1 \dots N_{n+1} \longrightarrow t_1 \dots t_n\ s$$

of G, the product grammar G_A will have all possible productions

$$N_{q_1 \dots q_n} \longrightarrow t'_1 \dots t'_n\ s'$$

where the m-contextual term word $t_1 \dots t_n$ is partially accepted by A (see Sect. 2) with initial control word $(q_1 \dots q_n)$ and final control word $(q'_1 \dots q'_m)$. Furthermore, $t'_1 \dots t'_n$ is obtained from $t_1 \dots t_n$ by pairing each of its m variables with

the associated component of the final control word, and s' is obtained from s by pairing each of its variables with a word on Q^*, so as to complete every instance of G_A's non-terminals according to (14). The result of this is that a pair (s,t) belongs to $L(G_A)$ if and only if $(s,t) \in L(G)$ and $s \in L$. In other words, the second projection of G_A is exactly the set of terms who are the image of some term in L by $\rightarrow_R^*$, so $\pi_2(L(G_A)) = \rightarrow_R^* (L)$. By forgetting the left projection of every grammar production, one gets a grammar where all non-terminals have an arity of 1. Such grammars are called *regular tree grammars* and generate recognizable term languages. In this case, we obtain a regular grammar generating $\rightarrow_R^* (L)$. □

A.2 Derivation of Suffix Systems

Lemma 6.2 *For any suffix term rewriting system R,*

$$s \xrightarrow[R]{k} t,\ k>0 \iff \begin{cases} \exists\ \bar{s},\bar{t} \in T(F,X),\ \sigma,\tau \in S(F,X),\ k'>0, \\ \qquad s = \bar{s}\sigma\ \wedge\ t = \bar{t}\tau \wedge\ \bar{s} \xrightarrow[R]{k'} \bar{t} \\ \wedge\ \forall x \in Var(\bar{s}) \cap Var(\bar{t})\ \sigma(x) \xrightarrow[R]{\leq k-k'} \tau(x). \end{cases}$$

Proof. First note that, since $\rightarrowtail_R\ \subseteq \rightarrow_R$, the inverse implication is trivial. It only remains to prove the direct implication. We will reason by induction on k:

- $k=1$: $s \rightarrow_R t$ implies that there is a context c and a substitution σ such that $s = c[l\sigma]$ and $t = c[r\sigma]$. Thus by definition of suffix rewriting $c[l] \rightarrowtail_R c[r]$, and of course for all variable x, $\sigma(x) \rightarrow_R^0 \sigma(x)$.
- $k \Rightarrow k+1$: let $s \rightarrow_R^k s' \xrightarrow[l,r]{}_p t$ with lRr, $s = \bar{s}\sigma$ and $s' = \bar{s'}\sigma'$. Two cases:
  - $p \in Pos(\bar{s'})$: if there exists a context c such that $\bar{s'} = c[l]$, then we have $\bar{s} \rightarrow_R^k \bar{s'} \rightarrowtail_R c[r]$ and the condition is verified with $k' = k$ and $t = (c[r])\sigma'$. If not, then there must exist a context c and a non-trivial substitution ω' such that $\bar{s'} = (c[l])\omega'$. As R is suffix and as, by induction hypothesis, $\bar{s} \rightarrowtail_R^* \bar{s'}$, there must exist ω such that $\bar{s} = (c[l])\omega$ and for all variable x common to l and r, $\omega(x) \rightarrowtail_R^* \omega'(x)$. We can then write s as $(c[l])\omega\sigma$, t as $(c[r])\omega'\sigma'$, and verify the condition is true with $k' = 1$.
  - $p \notin Pos(\bar{s'})$: by induction hypothesis, one can find $k' > 0$ such that for all x common to $\bar{s}$ and $\bar{s'}$, $\sigma(x) \rightarrow_R^{\leq k-k'} \sigma'(x)$. Furthermore, by applying rule (l,r) to one of the $\sigma'(x)$, we get $\sigma'(x) \rightarrow_{l,r} \tau(x)$. We thus have $t = \bar{s'}\tau$, $\bar{s} \rightarrowtail_R^{k'} \bar{s'}$ and $\sigma(x) \rightarrow_R^{k-k'+1} \tau(x)$ for all x in both $\bar{s}$ and $\bar{s'}$, which verifies the condition and concludes the proof. □

Lemma 6.3. *For all recognizable linear term rewriting system R over F and X, there exist a finite ranked alphabet Q and three finite rewriting systems*

- $R_+ \subseteq \{px \rightarrow fp_1x_1 \dots p_nx_n \mid f \in F,\ p,p_1,\dots,p_n \in Q,\ x,x_1,\dots,x_n \in X^*\}$,

– $R_{=} \subseteq \{px \to qy \mid p, q \in Q,\ x, y \in X^*\}$,
– $R_{-} \subseteq \{fp_1x_1 \dots p_nx_n \to px \mid f \in F,\ p, p_1, \dots, p_n \in Q,\ x, x_1, \dots, x_n \in X^*\}$

such that

$$s \xrightarrow[R]{*} t \iff s \xrightarrow[R_+ \cup R_=]{*} \circ \xrightarrow[R_- \cup R_=]{*} t.$$

Proof. Let R be a recognizable linear rewriting system over F and X. Let $(A_i, B_i)_{i \in [1,l]}$ be l pairs of finite top-down automata, each accepting a set of rules of R. The set of states of A_i (resp. B_i) will be referred to as P_i (resp. Q_i). Without losing generality, we will suppose that all P_i and Q_i are pairwise disjoint. We also define P as the union of all P_i and Q as the union of all Q_i. For all state $p \in P \cup Q$, define $\nu(p)$ as the set of all possible variable boundaries in the language accepted with initial state p. The way we defined recognizable linear systems, i.e. with a finite number of variables, we can consider without losing generality that $|\nu(p)| = 1$ for all p.

In a first step, we define a new rewriting system R' on $F \cup P \cup Q$ and X. The state alphabets P and Q are considered as ranked alphabets, where the arity of any of their symbols is equal to the number of variables appearing in the terms of its associated term language (which can be supposed unique without losing generality). For instance, suppose that from some state p, automaton A accepts the language g^*fxy. Then, p will be considered as a binary symbol.

We give R' the following set of rules. For all rule $pf \to p_1 \dots p_m$ in some A_i such that $\nu(p) = \nu(p_1) \dots \nu(p_m) = x_1 \dots x_n$, we have:

$$fp_1\nu(p_1) \dots p_m\nu(p_m) \ \to \ px_1 \dots x_n \ \in \ R'.$$

Rules of this kind allow us to consume a left-hand side of a rule in the input tree. For all rule $qf \to q_1 \dots q_m$ in some B_i such that $\nu(q) = \nu(q_1) \dots \nu(q_m) = x_1 \dots x_n$, we have:

$$qx_1 \dots x_n \ \to \ fq_1\nu(q_1) \dots q_m\nu(q_m) \ \in \ R'.$$

Rules of this kind allow us to produce a right-hand side of a rule whose left-hand side has been previously consumed. Finally, for all pair (p_0, q_0) of initial states of some pair (A_i, B_i), with $\nu(p_0) = x_1 \dots x_n$ and $\nu(q_0) = x_{k_1} \dots x_{k_m}$ we have:

$$p_0x_1 \dots x_n \ \to \ q_0x_{k_1} \dots x_{k_m} \ \in \ R'.$$

This simulates the application of a rewrite rule from $L(A_i) \times L(B_i)$ by initiating a run of automaton B_i when a successful 'reverse run' of A_i has been achieved. When restricted to $T(F)^2$, the derivation of R' coincides with $\to_R^*$:

$$\forall s, t \in T(F),\ s \xrightarrow[R]{*} t \iff s \xrightarrow[R']{*} t. \tag{15}$$

The proof of this property is not difficult and will thus not be detailed here. In the rest of the proof, p, q and all variations thereof designate automata control

states in $P \cup Q$, variable words in X^* are denoted by $u, v, u_i, v_i, \ldots$, and σ is a variable renaming.

In a second step, we define R_- as $\{fp_1u_1 \ldots p_nu_n\, R'\, pu\}$ ('consuming' rules), R_+ as $\{qv\, R'\, fq_1v_1 \ldots q_nv_n\}$ ('producing' rules) and $R_=$ as the smallest binary relation in $T(F,X)^2$ closed by the following inference rules:

$$\frac{}{pu\ R_=\ pu} \quad (1) \qquad \frac{pu\ R'\ qv}{pu\ R_=\ qv} \quad (2)$$

$$\frac{pu\ R_=\ qv}{pu\sigma\ R_=\ qv\sigma} \quad (3) \qquad \frac{qu\ R_=\ q'v \quad q'v\ R_=\ q''z}{qu\ R_=\ q''z} \quad (4)$$

$$\frac{pu\ R_-\ fp_1u_1 \ldots p_nu_n \quad fq_1v_1 \ldots q_nv_n\ R_+\ qv \quad \forall i,\ p_iu_i\ R_=\ q_iv_i}{pu\ R_=\ qv} \quad (5)$$

Since F, P, Q and X are finite, and each symbol p of $P \cup Q$ has a definite arity, then $R_=$ is finite and effectively computable. Let us mention a simple property of $R_=$:

$$\forall\, pu, t, qv \in T(F,X),\ pu \xrightarrow[R_- \cup R_=]{*} t \xrightarrow[R_+ \cup R_=]{*} qv \implies pu\, R_=\, qv. \tag{16}$$

This can be proved by induction on the nesting depth k of term t:

$k = 0$: no rule of R_- or R_+ is applied, thus $pu \rightarrowtail^*_{R_=} qv$. Then, by inference rule (4), the property is true.

$k \Rightarrow k+1$: let us decompose the derivation sequence between s and t:

$$pu \xrightarrow[R_=]{} p'u' \xrightarrow[R_-]{} fp_1u_1 \ldots p_nu_n$$
$$\xrightarrow[R_- \cup R_=]{*} \circ \xrightarrow[R_+ \cup R_=]{*} fq_1v_1 \ldots q_nv_n \xrightarrow[R_+]{} q'v' \xrightarrow[R_=]{} qv.$$

By induction hypothesis, we know that $fp_1u_1 \ldots p_nu_n \rightarrowtail^*_{R_=} fq_1v_1 \ldots q_nv_n$, so by inference rule (5), $p'u'\, R_=\, q'v'$. Finally, by inference rule (4), $pu\, R_=\, qv$.

It remains to prove that R_+, $R_=$ and R_- verify the lemma. By (15), it suffices to show that $\rightarrowtail^*_{R_+ \cup R_=} \circ \rightarrowtail^*_{R_- \cup R_=} = \rightarrowtail^*_{R'}$. Proving the direct inclusion is equivalent to proving $R_= \subseteq \rightarrowtail^*_{R'}$, which can be seen easily by observing the inference rules given above. To prove the converse we will establish, by induction on k, the following property:

$$\forall k,\ s \xrightarrow[R']{k} t \implies \exists\, c \in C_1(F),\ (q_iu_i)_{i \in [n]} \in T(F,X),$$
$$s \xrightarrow[R_+ \cup R_=]{*} c[q_1u_1 \ldots q_nu_n] \xrightarrow[R_- \cup R_=]{*} t.$$

$k = 0$: $s = t = c$, $n = 0$, so trivially $s \rightarrowtail^*_{R_+ \cup R_=} c \rightarrowtail^*_{R_- \cup R_=} t$.

$k \Rightarrow k+1$: let $s \rightarrowtail^k_{R'} t \rightarrowtail_{l,r} t'$ with $l R' r$. By induction hypothesis,

$$s \rightarrowtail^*_{R_+ \cup R_=} c[q_1 u_1 \dots q_n u_n] \rightarrowtail^*_{R_- \cup R_=} t = c[t_1 \dots t_n].$$

There are three possible cases:

- $\exists\, c',\ t = c'[t_1 \dots t_n\, l]$: in this case s can be written $c'[s_1 \dots s_n\, l]$, so the following derivation sequence is valid:

$$\begin{aligned} s &\xrightarrow[R_+ \cup R_=]{*} c'[q_1 u_1 \dots q_n u_n\, l] \xrightarrow[R_+]{} c'[q_1 u_1 \dots q_n u_n\, q_l u] \\ &\xrightarrow[R_=]{} c'[q_1 u_1 \dots q_n u_n\, q_r v] \text{ (because } l\, R'\, r) \\ &\xrightarrow[R_-]{} c'[q_1 u_1 \dots q_n u_n\, r] \xrightarrow[R_- \cup R_=]{*} c'[t_1 \dots t_n\, r] = t'. \end{aligned}$$

- $\exists\, i,\ t_i = \bar{t_i}[l]$: the only way to produce t_i from $q_i u_i$ is along the steps

$$q_i u_i \rightarrowtail^*_{R_- \cup R_=} \bar{t_i}[q_l u] \rightarrowtail_{R_-} t_i.$$

  Thus the following derivation is valid:

$$\begin{aligned} s &\xrightarrow[R_+ \cup R_=]{*} c[q_1 u_1 \dots q_n u_n] \xrightarrow[R_- \cup R_=]{*} c[q_1 u_1 \dots \bar{t_i}[q_l u] \dots q_n u_n] \\ &\xrightarrow[R_=]{} c[q_1 u_1 \dots \bar{t_i}[q_r v] \dots q_n u_n] \xrightarrow[R_-]{} t'. \end{aligned}$$

- $\exists\, c',\ j > i \geq 1,\ t = c'[t_1 \dots t_i\, l\, t_{j+1} \dots t_n]$, with $l = \bar{l}[t_{i+1} \dots t_j]$. The derivation sequence between s and t' can then be written

$$\begin{aligned} s &\xrightarrow[R_+ \cup R_=]{*} c[q_1 u_1 \dots q_n u_n] \xrightarrow[R_- \cup R_=]{*} c[t_1 \dots t_n] \\ &\xrightarrow[R_+]{*} c[t_1 \dots t_i\, q'_{i+i} u'_{i+1} \dots q'_j u'_j\, t_{j+1} \dots t_n] \\ &\xrightarrow[R_+]{} c'[t_1 \dots t_i\, q_l u\, t_{j+1} \dots t_n] \xrightarrow[R_=]{} c'[t_1 \dots t_i\, q_r v\, t_{j+1} \dots t_n] \\ &\xrightarrow[R_-]{} c'[t_1 \dots t_i\, r\, t_{j+1} \dots t_n] = t' \end{aligned}$$

  Notice that for all $k \in [i+1, j]$, $q_k u_k \rightarrowtail^*_{R_- \cup R_=} t_k \rightarrowtail^*_{R_+ \cup R_=} q'_k u'_k$. Thus, by (16), $q_k u_k\, R_=\, q'_k u'_k$, hence

$$\begin{aligned} s &\xrightarrow[R_+ \cup R_=]{*} c[q_1 u_1 \dots q_n u_n] \xrightarrow[R_+ \cup R_=]{*} c'[q_1 u_1 \dots t_i\, q_l u\, q_{j+1} u_{j+1} \dots q_n u_n] \\ &\xrightarrow[R_- \cup R_=]{*} c'[t_1 \dots t_i\, r\, t_{j+1} \dots t_n] = t'. \end{aligned}$$

This concludes the proof of lemma 6.3. □

Proposition 6.5. *The image and inverse image of any recognizable term language by the derivation of a recognizable linear suffix term rewriting system is recognizable.*

Proof. Let R_0 be a recognizable linear suffix term rewriting system on $T(F)$, R an equivalent normalized system on $T(F \cup F')$, with F' a set of new function symbols. Let G be the grammar recognizing its derivation, built as in the proof to Theorem 6.4, $N \cup \{I, I'\}$ its set of non-terminals. Let A be a finite non-deterministic top-down tree automaton accepting a recognizable language L. Suppose Q_A is the set of control states of A, disjoint from F and X, $q_0 \in Q_A$ its unique initial state. Let Q'_A be a disjoint copy of Q_A, we define the following grammar G_A having non-terminals in $Q_A \cup Q'_A \cup (Q_A \times Ran(R_+ \cup R')^* | Dom(R_- \cup R')^*$ and the following set of production rules:

- For all rule $rf \to_A fr_1 \dots r_n$:

$$r \longrightarrow f(r_1)_1 \dots (r_n)_1 \times f(r_1)_2 \dots (r_n)_2 \tag{17}$$
$$r' \longrightarrow fr'_1 \dots r'_n \tag{18}$$

- For all $r \in Q_A$, $px \in Dom(R) \cap Ran(R)$:

$$r \longrightarrow (r, px)|px, \tag{19}$$

- For all rule $p_1x_1 \dots p_nx_n|v \longrightarrow p'_1x'_1 \dots p'_nx'_n|v'$ of type (4) in G and state word $r_1 \dots r_n \in Q_A^*$:

$$(r_1, p_1x_1) \dots (r_n, p_nx_n)|v \longrightarrow (r_1, p'_1x'_1) \dots (r_n, p'_nx'_n)|v' \tag{20}$$

- For all rule $u_1\, px\, u_2|v \to s_1(f\mu_{i+1} \dots \mu_j)s_2 \times t$ of type (5) of G and states word $r_1 \dots r_i\, r\, r_{j+1} \dots r_n \in Q_A^*$:

$$(r_1, p_1) \dots (r_n, p_n)|v \longrightarrow s' \times t' \tag{21}$$

  where $rf \to_A fr_{i+1} \dots r_j$ and s' and t' are obtained from s and t by replacing each occurrence of p_k in a non-terminal variable by (r_k, p_k).
- For all rule $p_1x_1 \dots p_nx_n|v \to s \times t$ of type (6) of G and word $r_1 \dots r_n \in Q_A^n$:

$$(r_1, p_1) \dots (r_n, p_n)|v \longrightarrow s' \times t' \tag{22}$$

  where s' and t' are obtained from s and t by replacing each occurrence of p_i in a non-terminal variable by (r_i, p_i).
- Finally, for all $xRpx$, $qxRx$, $x \in X$ and $r \in Q_A$:

$$(r, px)|qx \longrightarrow r \tag{23}$$
$$(r, px)| \longrightarrow r'. \tag{24}$$

One can see that, starting from non-terminal q_0, grammar G_A only accepts pairs (s, t) such that $s \to_R^* t$ and $s \in L$. Thus the set of all t such that (s, t) is generated by G_A from q_0 is exactly the image of L by the derivation of R:

$$L(G_A, q_0) = \xrightarrow[R]{*}(L).$$

An automaton recognizing this set of terms can be built by taking the right projection of G_A, and by treating each non-terminal variable as a unary non-terminal. The rules of this automaton are given by the rules of G_A broken into several rules over these non-terminals of length 1.

Note that this proof is totally symmetrical, and that the synchronization of G by a finite automaton A could be done on the second projection instead of the first. Thus the converse result. □